From Two-Way to One-Way Finite State Transducers
Emmanuel Filiot, Olivier Gauwin, Pierre-Alain Reynier, Frédéric Servais

To cite this version:
Emmanuel Filiot, Olivier Gauwin, Pierre-Alain Reynier, Frédéric Servais. From Two-Way to One-Way Finite State Transducers. 28th Annual ACM/IEEE Symposium on Logic in Computer Science, Jun 2013, New Orleans, United States. IEEE Computer Society, pp.468-477, 2013, <10.1109/LICS.2013.53>. <hal-00946161>

HAL Id: hal-00946161
https://hal.archives-ouvertes.fr/hal-00946161
Submitted on 13 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
From Two-Way to One-Way Finite State Transducers

Emmanuel Filiot
LACL
University Paris-Est Créteil

Olivier Gauwin
LaBRI,
University of Bordeaux

Pierre-Alain Reynier
LIF, Aix-Marseille Univ.
& CNRS, UMR 7279

Frédéric Servais
Hasselt University and
Transnational University of Limburg

Abstract—Any two-way finite state automaton is equivalent to some one-way finite state automaton. This well-known result, shown by Rabin and Scott and independently by Shepherdson, states that two-way finite state automata (even non-deterministic) characterize the class of regular languages. It is also known that this result does not extend to finite string transductions: (deterministic) two-way finite state transducers strictly extend the expressive power of (functional) one-way transducers. In particular deterministic two-way transducers capture exactly the class of MSO-transductions of finite strings.

In this paper, we address the following definability problem: given a function defined by a two-way finite state transducer, is it definable by a one-way finite state transducer? By extending Rabin and Scott’s proof to transductions, we show that this problem is decidable. Our procedure builds a one-way transducer, which is equivalent to the two-way transducer, whenever one exists.

I. INTRODUCTION

In formal language theory, the importance of a class of languages is often supported by the number and the diversity of its characterizations. One of the most famous example is the class of regular languages of finite strings, which enjoys, for instance, computational (automata), algebraic (syntactic congruence) and logical (monadic second order (MSO) logic with one successor) characterizations. The study of regular languages has been very influential and several generalizations have been established. Among the most notable ones are the extensions to infinite strings [1] and trees [2]. On finite strings, it is well-known that both deterministic and non-deterministic finite state automata define regular languages. It is also well-known that the expressive power of finite state automata does not increase when the reading head can move left and right, even in presence of non-determinism. The latter class is known as non-deterministic two-way finite state automata and it is no more powerful than (one-way) finite state automata. The proof of this result was first shown in the seminal paper of Rabin and Scott [3], and independently by Shepherdson [4].

The picture of automata models over finite strings changes substantially when, instead of languages, string transductions, i.e. relations from strings to strings, are considered. Transducers generalize automata as they are equipped with a one-way output tape. At each step they read an input symbol, they can append several symbols to the output tape. Their transition systems can be either deterministic or non-deterministic. Functional transducers are transducers that define functions instead of relations. For instance, deterministic transducers are always functional. In this paper, we are interested in transducers that define functions, but that can be non-deterministic.

As for automata, the reading head of transducers can move one-way (left-to-right) or two-way. (One-way) finite state transducers have been extensively studied [5], [6]. Non-deterministic (even functional) one-way transducers (NFTs) strictly extend the expressive power of deterministic one-way transducers (DFTs), because non-determinism allows one to express local transformations that depend on properties of the future of the input string.

Two-way finite state transducers define regular transformations that are beyond the expressive power of one-way transducers [7]. They can for instance reverse an input string, swap two substrings or copy a substring. The transductions defined by two-way transducers have been characterized by other logical and computational models. Introduced by Courcelle, monadic second-order definable transductions are transformations from graphs to graphs defined with the logic MSO [8]. Engelfriet and Hoogeboom have shown that the monadic second-order definable functions are exactly the functions definable by deterministic two-way finite state transducers (2DFTs) when the graphs are restricted to finite strings [9]. Recently, Alur and Černý have characterized 2DFT-definable transductions by a deterministic one-way model called streaming string transducers [10] and shown how they can be applied to the verification of list-processing programs [11]. Streaming string transducers extend DFTs with a finite set of output string variables. At each step, their content can be reset or updated by either prepending or appending a finite string, or the content of another variable, in a copyleft manner. Extending 2DFTs with non-determinism does not increase their expressive power when they define functions: non-deterministic two-way finite state transducers (2NFTs) that are functional define exactly the class of functions definable by 2DFTs [9], [12]. To summarize, there is a strict hierarchy between $DFT$, functional $NFT$- and $2DFT$-definable transductions.

Several important problems are known to be decidable for one-way transducers. The $functionality$ problem for $NFT$, decidable in $PTime$ [13], [14], asks whether a given $NFT$ is functional. The $determinizability$ problem, also decidable in $PTime$ [15], [14], asks whether a given functional $NFT$ can be determined, i.e. defines a subsequential function. Subsequent functions are those functions that can be defined by $DFT$s equipped with an additional output function from final states to finite strings, which is used to append a last string.
to the output when the computation terminates successfully in some final state. Over strings that always end with a unique end marker, subsequential functions are exactly the functions definable by DFTs. For 2NFTs, the functionality problem is known to be decidable [16]. Therefore the determinizability problem is also decidable for 2NFTs, since functional 2NFTs and DFTs have the same expressive power. In the same line of research, we address a definability problem in this paper. In particular we answer the fundamental question of NFT-definability of transductions defined by functional 2NFTs.

**Theorem 1.** For all functional 2NFTs $T$, it is decidable whether the transduction defined by $T$ is definable by an NFT.

The proof of Theorem 1 extends the proof of Rabin and Scott [3] from automata to transducers$^1$. The original proof of Rabin and Scott is based on the following observation about the runs of two-way automata. Their shapes have a nesting structure: they are composed of many zigzags, each zigzag being itself composed of simpler zigzags. Basic zigzags are called $z$-motions as their shapes look like a $Z$. Rabin and Scott proved that for automata, it is always possible to replace a $z$-motion by a single pass. Then from a two-way automaton $A$, it is possible to construct an equivalent two-way automaton $B$ (called the squeeze of $A$) which is simpler in the following sense: accepting runs of $B$ are those of $A$ in which some $z$-motions have been replaced by single pass runs. Last, they argue$^2$ that after a number of applications of this construction that depends only on the number of states of $A$, every zigzag can be removed, yielding an equivalent one-way automaton.

The extension to 2NFTs faces the following additional difficulty: it is not always possible to replace a $z$-motion of a transducer by a single pass. Intuitively, this is due to the fact that 2NFTs are strictly more expressive than NFTs. As our aim is to decide when a 2NFT $T$ is NFT-definable, we need to prove that the NFT-definability of $T$ implies that of every $z$-motion of $T$, to be able to apply the squeeze construction. The main technical contribution of this paper is thus the study of the NFT-definability of $z$-motions of transducers. We show that this problem is decidable, and identify a characterization which allows one to prove that the NFT-definability of $T$ implies that of every $z$-motion of $T$.

This characterization expresses requirements about the output strings produced along loops of $z$-motions. We show that when $z$-motions are NFT-definable, the output strings produced by the three passes on a loop are not arbitrary, but conjugates. This allows us to give a precise characterization of the form of these output strings. We show that it is decidable to check whether all outputs words have this form. Last, we present how to use this characterization to simulate an NFT-definable $z$-motion by a single pass.

**Applications** By Theorem 1 and since functionality is decidable for 2NFTs, it is also decidable, given a 2NFT, whether the transduction it defines is definable by a functional NFT. Another corollary of Theorem 1 and the fact that functionality of 2NFTs and determinizability of NFTs are both decidable is the following theorem:

**Theorem 2.** For all 2NFTs $T$, it is decidable whether the transduction defined by $T$ is a subsequential function.

A practical application of this result lies in the static analysis of memory requirements for evaluating (textual and functional) document transformations in a streaming fashion. In this scenario, the input string is received as a left-to-right stream. When the input stream is huge, it should not be entirely loaded in memory but rather processed on-the-fly. Similarly, the output string should not be stored in memory but produced as a stream. The remaining amount of memory needed to evaluate the transformation characterizes its streaming space complexity. Streamable transformations are those transformations for which the required memory is bounded by a constant, and therefore is independent on the length of the input stream. It is known that streamable transformations correspond to transformations definable by subsequential (functional) NFTs [18]. The streamability problem asks, given a transformation defined by some transducer, whether it is streamable. Therefore for transformations defined by functional NFTs, streamability coincides with determinizability, and is decidable in PTime [15], [14]. Theorem 2 is a generalization of this latter result to regular transformations, i.e. transformations defined by functional 2NFTs, MSO transducers or streaming string transducers [10]. Other streamability problems have been studied for XML validation [19], [20], XML queries [21] and XML transformations [18]. However the XML tree transformations of [18] are incomparable with the regular string transformations studied in this paper.

**Related work** Most of the related work has already been mentioned. To the best of our knowledge, it is the first result that addresses a definability problem between two-way and one-way transducers. In [22], two-way transducers with a two-way output tape are introduced with a special output policy: each time a cell at position $i$ of the input tape is processed, the output is written in the cell at position $i$ of the output tape. With that restriction, it is shown that two-way and one-way transducers (NFTs) define the same class of functions. In [23], the result of Rabin and Scott, and Shepherdson, is extended to two-way automata with multiplicities. In this context, two-way automata strictly extend one-way automata.

**Organization of the paper** Section II introduces necessary preliminary definitions. In Section III, we describe the general decision procedure for testing NFT-definability of functional 2NFTs. We introduce $z$-motion transductions induced by 2NFTs and show that their NFT-definability is necessary. The decidability of this necessary condition as well as the construction from $z$-motion transducers to NFTs are the most technical results of this paper and are the subject of Section IV. We finally discuss side results and further questions in Section V.
All technical details and omitted proofs can be found in the full version of this paper [24].

II. ONE-WAY AND TWO-WAY FINITE STATE MACHINES

Words, Languages and Transductions Given a finite alphabet $\Sigma$, we denote by $\Sigma^*$ the set of finite words over $\Sigma$, and by $\epsilon$ the empty word. The length of a word $u \in \Sigma^*$ is its number of symbols, denoted by $|u|$. For all $i \in \{1, \ldots, |u|\}$, we denote by $u[i]$ the $i$-th letter of $u$. Given $1 \leq i \leq j \leq |u|$, we denote by $u[i..j]$ the word $u[i]u[i+1] \ldots u[j]$ and by $u[j..i]$ the word $u[j]u[j-1] \ldots u[i]$. We say that $v \in \Sigma^*$ is a factor of $u$ if there exist $u_1, u_2 \subseteq \Sigma^*$ such that $u = u_1v_2u_2$. By $\pi(u)$ we denote the mirror of $u$, i.e. the word of length $|u|$ such that $\pi[i] = u[|u| - i + 1]$ for all $1 \leq i \leq |u|$.

The primitive root of $u \in \Sigma^*$ is the shortest word $v$ such that $u = v^k$ for some integer $k \geq 1$, and is denoted by $\mu(u)$. Two words $u$ and $v$ are conjugates, denoted by $\sim$, if there exist $x, y \in \Sigma^*$ such that $u = xy$ and $v = yx$, i.e. $u$ can be obtained from $v$ by a cyclic permutation. Note that $\sim$ is an equivalence relation. We will use this fundamental lemma:

Lemma 1 ([25]). Let $u, v \in \Sigma^*$. If there exists $n \geq 0$ such that $u^n$ and $v^n$ have a common factor of length at least $|u| + |v| - \gcd(|u|, |v|)$, then $\mu(u) \sim \mu(v)$.

Note that if $\mu(u) \sim \mu(v)$, then there exist $x, y \in \Sigma^*$ such that $u \in (xy)^*$ and $v \in (yx)^*$.

A language over $\Sigma$ is a set $L \subseteq \Sigma^*$. A transduction over $\Sigma$ is a relation $R \subseteq \Sigma^* \times \Sigma^*$. Its domain is denoted by $\text{dom}(R)$, i.e. $\text{dom}(R) = \{u \mid \exists v, (u, v) \in R\}$, while its image $\{v \mid \exists u, (u, v) \in R\}$ is denoted by $\text{img}(R)$. A transduction $R$ is functional if it is a function.

Automata A non-deterministic two-way finite state automaton $3$ (2NFA) over a finite alphabet $\Sigma$ is a tuple $A = (Q, q_0, F, \Delta)$ where $Q$ is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and $\Delta$ is the transition relation, of type $\Delta \subseteq Q \times \Sigma \times Q \times \{+1, -1\}$. It is deterministic if for all $(p, a) \in Q \times \Sigma$, there is at most one pair $(q, m) \in Q \times \{+1, -1\}$ such that $(p, a, q, m) \in \Delta$. In order to see how words are evaluated by $A$, it is convenient to see the input as a right-infinite input tape containing the word (starting at the first cell) followed by blank symbols. Initially the head of $A$ is on the first cell in state $q_0$ (the cell at position 1). When $A$ reads an input symbol, depending on the transitions in $\Delta$, its head moves to the left ($-1$) if the head was not in the first cell, or to the right ($+1$) and changes its state. $A$ stops as soon as it reaches a blank symbol (therefore at the right of the input word), and the word is accepted if the current state is final.

A configuration of $A$ is a pair $(q, i) \in Q \times (\mathbb{N} - \{0\})$ where $q$ is a state and $i$ is a position on the input tape. A run $\rho$ of $A$ is a finite sequence of configurations. The run $\rho = (p_1, i_1) \ldots (p_m, i_m)$ is a run on an input word $u \in \Sigma^*$ of length $n$ if $p_1 = q_0$, $i_1 = 1$, $i_m \leq n + 1$, and for all $k \in \{1, \ldots, m - 1\}$, $1 \leq i_k \leq n$ and $(p_k, u[i_k], p_{k+1}, i_{k+1} - i_k) \in \Delta$. It is accepting if $i_m = n + 1$ and $p_m \in F$. The language of a 2NFA $A$, denoted by $L(A)$, is the set of words $u$ such that there exists an accepting run of $A$ on $u$.

A non-deterministic (one-way) finite state automaton (NFA) is a 2NFA such that $\Delta \subseteq Q \times \Sigma \times Q \times \{+1\}$, therefore we will often omit $\Delta$ as a subset of $Q \times \Sigma \times Q$. Any 2NFA is effectively equivalent to an NFA. It was first proved by Rabin and Scott, and independently by Shepherdson [3], [4].

Transducers Non-deterministic two-way finite state transducers (2NFTs) over $\Sigma$ extend NFAs with a one-way left-to-right output tape. They are defined as 2NFAs except that the transition relation $\Delta$ is extended with outputs: $\Delta \subseteq Q \times \Sigma \times Q \times \{-1, +1\}$. If a transition $(q, a, v, q', m)$ is fired on a letter $a$, the word $v$ is appended to the right of the output tape and the transducer goes to state $q'$. With we assume that for all $p, q \in Q, a \in \Sigma$ and $m \in \{+1, -1\}$, there exists at most one $v \in \Sigma^*$ such that $(p, a, v, q, m) \in \Delta$. We also denote $v$ by $\text{out}(p, a, q, m)$.

A run of a 2NFT is a run of its underlying automaton, i.e. the 2NFAs obtained by ignoring the output. A run $\rho$ may be simultaneously a run on a word $u$ and on a word $u' \neq u$. However, when the underlying input word is given, there is a unique sequence of transitions associated with $\rho$. Given a 2NFT $T$, an input word $u \in \Sigma^*$ and a run $\rho = (p_1, i_1) \ldots (p_m, i_m)$ of $T$ on $u$, the output of $\rho$ on $u$, denoted by $\text{out}_u(\rho)$, is the word obtained by concatenating the outputs of the transitions followed by $\rho$, i.e. $\text{out}_u(\rho) = \text{out}(p_1, u[i_1], p_2, i_2 - i_1) \ldots \text{out}(p_{m-1}, u[i_{m-1}], p_m, i_m - i_{m-1})$.

If $\rho$ contains a single configuration, we let $\text{out}_u(\rho) = \epsilon$. When the underlying input word $u$ is clear from the context, we may omit the exponent $u$. The transduction defined by $T$ is the relation $R(T) = \{(u, \text{out}_u(\rho)) \mid \rho$ is an accepting run of $T$ on $u\}$. We may often just write $T$ when it is clear from the context. A 2NFT $T$ is functional if the transduction it defines is functional. The class of functional 2NFAs is denoted by $f2NFT$. In this paper, we mainly focus on $f2NFTs$. The domain of $T$ is defined as $\text{dom}(T) = \text{dom}(R(T))$. The domain $\text{dom}(T)$ is a regular language that can be defined by the 2NFA obtained by projecting away the output part of the transitions of $T$, called the underlying input automaton. A deterministic two-way finite state transducer (2DFT) is a 2NFT whose underlying input automaton is deterministic. Note that 2DFTs are always functional, as there is at most one accepting run per input word. A non-deterministic (one-way) finite state transducer (NFT) is a 2NFT whose underlying automaton is an NFA$^4$. It is deterministic (written DFT) if the underlying automaton is a DFA.

We say that two transducers $T, T'$ are equivalent, denoted by $T \equiv T'$, whenever they define the same transduction, i.e. $R(T) = R(T')$. For all transducer classes $C$, we say that a transduction $R \subseteq \Sigma^* \times \Sigma^*$ is $C$-definable if there exists $T \in C$.

$^4$This definition implies that there is no $\epsilon$-transitions that can produce outputs, which may cause the image of an input word to be an infinite language. Those NFTs are sometimes called real-time in the literature.
such that $R = R(T)$. Given two classes $C, C'$ of transducers, and a transducer $T \in C$, we say that $T$ (effectively) $C'$-definable if one can construct an equivalent transducer $T' \in C'$. The $(C, C')$-definability problem takes as input a transducer $T \in C$ and asks to decide whether $T$ is $C'$-definable. If so, one may want to construct an equivalent transducer $T' \in C'$. In this paper, we prove that $(f2NFT, NFT)$-definability is decidable.

It is known that whether an $NFT$ $T$ is functional can be decided in PTime [13]. The class of functional $NFT$s is denoted by $fNFT$. Functional $NFT$s are strictly more expressive than $DFT$s. For instance, the function that maps any word $u \in \{a, b\}^+$ to $a^{|u|}$ if $|u| = a$, and to $b^{|u|}$ otherwise, is $fNFT$-definable but not $DFT$-definable. This result does not hold for $2NFT$s: functional $2NFT$s and $2DFT$s define the same class of transductions (Theorem 22 of [9]).

**Examples** Let $\Sigma = \{a, b\}$ and $\# \notin \Sigma$, and consider the transductions

1. $R_0 = \{(u, a^{|u|}) \mid u \in \Sigma^*, |u| = a\}$
2. $R_1 = \{(u, b^{|u|}) \mid u \in \Sigma^*, |u| = b\} \cup R_0$
3. $R_2 = \{a\# \# \#, \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \# \#}
If $T$ is a 2NFA, it is possible to construct a new automaton denoted by \textit{squeeze}(T) such that, for all accepting runs $\rho$ of $T$ on some input word $u$, there exists a “simpler” accepting run of \textit{squeeze}(T) on $u$, obtained from $\rho$ by replacing some $z$-motions by one-way runs that simulate three passes in parallel. It is illustrated by Fig. 1. For instance at the first step, there are two $z$-motions from $q_1$ to $q_2$ and from $q_3$ to $q_4$ respectively. Applying \textit{squeeze}(T) consists in non-deterministically guessing those $z$-motions and simulating them by one-way runs. This is done by the NFA $R_{T}(q_1, q_2)$ and $R_{T}(q_3, q_4)$ respectively. Depending on whether the $z$-motions enter from the left or the right, $z$-motions are replaced by runs of NFAs $R_{T}(\ldots)$ (that read the input backwardly) or $L_{T}(\ldots)$, as illustrated by the second iteration of \textit{squeeze} on Fig. 1.

An $N$-crossing run $\rho$ can be simplified into a one-way run after a constant number of applications of \textit{squeeze}. This result is unpublished so we prove it in this paper. In particular, we show that if $\rho$ is $N$-crossing, then its zigzag nesting depth decreases after $N$ steps. Moreover, if $\rho$ is $N$-crossing, then its zigzag nesting depth is also bounded by $N$. Therefore after $N^2$ applications of \textit{squeeze}, $\rho$ is transformed into a simple one-way run. It is sufficient to prove those results at the level of integer sequences. In particular, one can define \textit{squeeze}(s) the set of sequences obtained from a one-step sequence $s$ by replacing some $z$-motions of $s$ by strictly increasing or decreasing subsequences (for more details see [24]).

\textbf{Lemma 2.} Let $s$ be an $N$-crossing one-step sequence over $\{1, \ldots, m\}$. Then $1, 2, \ldots, m$ is in $\textit{squeeze}^{N^2}(s)$.

At the automata level, it is known that for all words $u$ accepted by a 2NFA $T$ with $N$ states, there exists an $N$-crossing accepting run on $u$. Therefore it suffices to apply \textit{squeeze} $N^2$ times to $T$. One gets an equivalent 2NFA $T^*$ from which the backward transitions can be removed while preserving equivalence with $T^*$, and so $T$.

\textbf{B. Extension to transducers: overview}

The construction used to show decidability of $NFT$-definability of $f2NFT$ follows the same ideas as Rabin and Scott’s construction. The main difference relies in the transformation of the local transducers defined by $z$-motion runs (that we call $ZNFTs$) into $NFTs$. Our procedure is built over a $ZNFT$-to-$NFT$ procedure. It is seen as a black-box in this section, but is the subject of the next section.

Compared to two-way automata, one faces an extra difficulty caused by the fact that $2NFTs$ (and $ZNFTs$) are not always $NFT$-definable. Therefore one defines a necessary condition that has to be tested each time we want to apply \textit{squeeze}. Let us consider again Fig. 1 when $T$ is a 2NFT. One defines from $T$ the transductions induced by local $z$-motion runs from a starting state $q_1$ to an ending state $q_2$, and show that those local transductions must be $NFT$-definable.

Once this necessary condition is satisfied, the construction \textit{squeeze} can be applied and works as for Rabin and Scott’s construction: the new transducer \textit{squeeze}(T) simulates $T$ and non-deterministically may guess that the next zigzag of $T$ is a $z$-motion run from some state $q_1$ to some state $q_2$, and thus can be simulated by a run of some $NFT$ $R_{T}(q_1, q_2)$ or $L_{T}(q_1, q_2)$, depending on whether it enters from the left or the right. Then \textit{squeeze}(T) switches to $R_{T}(q_1, q_2)$ (if it entered from the right) and once $R_{T}(q_1, q_2)$ reaches an accepting state, it may come back to its normal mode.

\textbf{C. $z$-motion transducers}

\textit{z}-motion transducers are defined like $2NFTs$ except that they must define \textit{functions} and to be accepting, a run on a word of length $n$ must be of the form $\rho(q_f, n + 1)$ where $\rho$ is a $z$-motion run and $q_f$ is an accepting state. Note that it implies that shape($\rho$) is always of the form $1, \ldots, n, n-1, \ldots, 1, \ldots, n$. The class of $z$-motion transducers is denoted by $ZNFTs$. Note that $z$-motion transducers are incomparable with $f2NFTs$. Indeed, $z$-motion transducers can define the transduction $u \in \Sigma^* \rightarrow \pi$, which is not $f2NFT$-definable as there are no end markers.

Let $T \in ZNFT$ and $\rho = \rho_1(q_1, 1) \ldots \rho_n(q_n, n)$ be a run of $T$ on a word of length $n$. We let $q_n = p_n$ and $r_1 = q_1$ and define the following shortcuts: for $1 \leq i \leq j \leq n$, out$_i[i, j] = \text{out}((p_1, i)(p_2, j), \ldots)$, and out$_2[i, j] = \text{out}((q_1, j) \ldots (q_i, i))$ and out$_3[i, j] = \text{out}((r_i, i) \ldots (r_j, j))$, and out$_3[i, n + 1] = \text{out}((r_i, i) \ldots (r_{n+1}, n + 1))$.

We characterize the $NFT$-definability of a $ZNFT$ by a property that we prove to be decidable. Intuitively, this property
Definition 1 (P-property). Let $T$ be a ZNFT. We say that $T$ satisfies the property $P$, denoted by $T \models P$, if for all words $u \in \text{dom}(T)$, for all accepting runs $\rho$ on $u$, and for all pairs of loops $(i_1, j_1)$ and $(i_2, j_2)$ of $\rho$ such that $j_1 \leq i_2$, there exist $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5 \in \Sigma^*$, $f, g : \mathbb{N}^2 \rightarrow \Sigma^*$ and constants $c_1, c_1', c_2, c_2' \geq 0$ such that $c_1, c_2, c_2' \geq 0$ and for all $k_1, k_2 \geq 0$,

$$f(k_1, k_2)x_0v_1^{n_1}x_1w_1^{n_2}x_2w_2^{n_3}x_3v_3^{n_4}x_4w_4^{n_5}x_5w_5^{n_6}x_6g(k_1, k_2) = \beta_1\beta_2^{c_1}\beta_3\beta_4^{c_2}\beta_5,$$

where $\eta_i = k_ic_i + c_i'$, $i \in \{1, 2\}$, and $x_i's, v_i's$ and $w_i's$ are words defined as depicted in Fig. 2.

The following key lemma is proved in Section IV.

Lemma 3. Let $T \in \text{ZNFT}$. $T \models P$ iff $T$ is NFT-definable. Moreover, $P$ is decidable and if $T \models P$, one can (effectively) construct an equivalent NFT.

Definition 2 ($\ell$-motion transductions induced by a f2NFT). Let $T = (Q, q_0, F, \Delta)$ be a f2NFT and $q_1, q_2 \in Q$. The transduction $L_T(q_1, q_2)$ (resp. $R_T(q_1, q_2)$) is defined as the set of pairs $(u_2, v_2)$ such that there exist $u \in \Sigma^*$, two positions $i_1 < i_2$ (resp. $i_2 < i_1$), an accepting run $\rho$ of $T$ on $u$ which can be decomposed as $\rho = \rho_1(q_1, i_1)\rho_2(q_2, i_2)\rho_3$ such that $u_2 = u[i_1 \ldots i_2]$ and

- $(q_1, i_1)\rho_2(q_2, i_2)$ is a $\ell$-motion run
- out($((q_1, i_1)\rho_2(q_2, i_2)) = v_2$

$\ell$-motions can be of two forms: either they start from the left and end to the right, or start from the right and end to the left. In order to avoid considering these two cases each time, we introduce the notation $\overline{T}$ that denotes the mirror of $T$: it is $T$ where the moves $+1$ are replaced by $-1$ and the moves $-1$ by $+1$. Moreover, the way $\overline{T}$ reads the input tape is slightly modified: it starts in position $n$ and a run is accepting if it reaches position $0$ in some accepting state. All the notions defined for NFTs carry over to their mirrors. In particular, $(u, v) \in R(T)$ iff $(\overline{v}, \overline{u}) \in R(\overline{T})$. The $\ell$-motion transductions $R_T(q_1, q_2)$ and $L_T(q_1, q_2)$ are symmetric in the following sense: $R_T(q_1, q_2) = L_T(\overline{T}, q_2)$ and $L_T(q_1, q_2) = R_T(q_1, \overline{T})$.

Proposition 2. The transductions $R_T(q_1, q_2)$ and $L_T(q_1, q_2)$ are NFT-definable.

Proof: We only consider the case $L_T(q_1, q_2)$, the other case being solved by using the equality $R_T(q_1, q_2) = L_T(\overline{T}, q_2)$. We first construct from $T$ a ZNFT $Z_T(q_1, q_2)$ which is like $T$ but its initial state is $q_1$, and it can move to an accepting state whenever it is in $q_2$. However $Z_T(q_1, q_2)$ may define input/output pairs $(u_2, v_2)$ that cannot be embedded into some pair $(u, v) \in R(T)$ as required by the definition of $L_T(q_1, q_2)$. Based on Shepherdson’s construction, we modify $Z_T(q_1, q_2)$ in order to take this constraint into account.

In the next subsection, we show that $R_T(q_1, q_2)$ and $L_T(q_1, q_2)$ must necessarily be NFT-definable for $T$ to be NFT-definable. For that purpose, it is crucial in Definition 2 to make sure that the $\ell$-motion $(q_1, i_1)\rho_2(q_2, i_2)$ can be embedded into a global accepting run of $T$. Without that restriction, it might be the case that $L_T(q_1, q_2)$ or $R_T(q_1, q_2)$ is not NFT-definable although the 2NFT $T$ is. Indeed, the domain of $L_T(q_1, q_2)$ or $R_T(q_1, q_2)$ would be too permissive and accept words that would be otherwise rejected by other passes of global runs of $T$. This is another difficulty when lifting Rabin and Scott’s proof to transducers, as for automata, the context in which a $\ell$-motion occurs is not important.

D. Decision procedure and proof of Theorem 1

We show that the construction squeeze(T) can be applied if the following necessary condition is satisfied.

Lemma 4. If $T$ is NFT-definable, then so are the transductions $R_T(q_1, q_2)$ and $L_T(q_1, q_2)$ for all states $q_1, q_2$. Moreover, it is decidable whether the transductions $R_T(q_1, q_2)$ and $L_T(q_1, q_2)$ are NFT-definable.

Sketch of proof: We have seen in Lemma 3 that NFT-definability of an ZNFT is characterized by Property $P$. Let $Z \in \text{ZNFT}$ that defines $L_T(q_1, q_2)$ for some $q_1, q_2$, we thus sketch the proof that $Z \models P$.

Consider two loops $(i_1, j_1)$, $(i_2, j_2)$ of a run of $Z$ on some word $u$, as in the premises of Property $P$. They induce a decomposition of $u = u_1u_2u_3u_4u_5$ with $u_2 = u[i_1 \ldots j_1 - 1]$ and $u_4 = u[i_2 \ldots j_2 - 1]$. By definition of the transduction $L_T(q_1, q_2)$, any word in $\text{dom}(Z)$ can be extended into a word in $\text{dom}(T)$. By hypothesis, $T$ is NFT-definable, thus there exists an equivalent NFT $T'$. As $T'$ has finitely many states, it is possible, by iterating the loops $(i_1, j_1)$ and $(i_2, j_2)$, to identify an input word of the form

$$u' = \alpha u_1u_2^{\epsilon_1}u_3u_4^{\epsilon_1'}u_5^{\epsilon_1}u_4^{\epsilon_1'}u_3^{\epsilon_1}u_2^{\epsilon_1}u_1^{\epsilon_1}$$

and a run $\rho'$ of $T'$ on this word which has two loops on the input subwords $u_2^{\epsilon_1}$ and $u_4^{\epsilon_1'}$. It is then easy to conclude.

Construction of squeeze(T) Assuming that the necessary condition is satisfied, we now explain how to construct the f2NFT squeeze(T). By hypothesis, the transductions $L_T(q_1, q_2)$ and $R_T(q_1, q_2)$ are NFT-definable for all $q_1, q_2$ by NFT $L_T(q_1, q_2)$ and $R_T(q_1, q_2)$ respectively (they exist by Proposition 2 and Lemma 3). As already said before, the main idea to define squeeze(T) is to non-deterministically (but repeatedly) apply $L_T(q_1, q_2)$, $R_T(q_1, q_2)$, or $T$, for some $q_1, q_2 \in Q$. However when applying $R_T(q_1, q_2)$, the head of squeeze(T) should move from the right to the left, so that we have to mirror the transitions of $R_T(q_1, q_2)$.

The transducer squeeze(T) has two modes, Z-mode and T-mode. In T-mode, it works as $T$ until it non-deterministically decides that the next zigzag is a $\ell$-motion from some state $q_1$ to some state $q_2$. Then it goes in Z-mode and runs $L_T(q_1, q_2)$ or $R_T(q_1, q_2)$, in which transitions to an accepting state have been replaced by transitions from $q_2$ in $T$, so that squeeze(T) returns in T-mode. From those transitions we also add transitions from the initial states of $L_T(q_2, q_3)$ and $R_T(q_2, q_5)$ for all $q_3 \in Q$, in case squeeze(T) guesses that
the next \( z \)-motion starts immediately at the end of the previous \( z \)-motion (for more details see [24]).

**Proposition 3.** Let \( T \in fNFT \) such that \( T \) is NFT-definable. Then \( \text{squeeze}(T) \) is defined and equivalent to \( T \).

Let \( T \in fNFT \). If \( T \) is NFT-definable, then the operator \( \text{squeeze} \) can be iterated on \( T \) while preserving equivalence with \( T \), by the latter proposition. By Proposition 1 \( T \) is \( N \)-crossing, and therefore, based on Lemma 2, it suffices to iterate \( \text{squeeze} \) \( N^2 \) times to remove all zigzags from accepting runs of \( T \), as stated by the following lemma:

**Lemma 5.** Let \( T \) be a fNFT with \( N \) states. If \( T \) is \( N \)-definable, then \( \text{squeeze}^N(T) \) is defined and equivalent to \( T \), and moreover, for all \((u, v) \in R(T)\), there exists an accepting run \( \rho \) of \( \text{squeeze}^N(T) \) on \( u \) such that \( \text{out}(\rho) = v \) and \( \rho \) is made of forward transitions only.

**Proof of Theorem 1** In order to decide whether a fNFT \( T \) is NFT-definable, it suffices to test whether \( \text{squeeze} \) can be applied \( N^2 \) times. More precisely, it suffices to set \( T_0 = T \), \( i = 0 \), and, while \( T_i \) satisfies the necessary condition (which is decidable by Lemma 4) and \( i \leq N^2 \), to increase \( i \) and set \( T_i = \text{squeeze}(T_{i-1}) \). If the procedure exits the loops before reaching \( N^2 \), then \( T \) is not NFT-definable, otherwise it is NFT-definable by the NFT construction when \( T \) is satisfied.

We first prove that Property \( \mathcal{P} \) is a necessary condition for NFT-definability. To prove the converse, we proceed in two steps. First, we define a procedure that tests whether a given ZNFT \( T \) is equivalent to a ZNFT that does not output anything on its backward pass (called \( \epsilon \text{ZNFT} \)), and then define another procedure that tests whether the latter ZNFT is equivalent to an NFT. We show that it is always true whenever \( T \models \mathcal{P} \). This approach is depicted in Fig. 3. The two steps are similar, therefore we mainly focus on the first step.

**A. Property \( \mathcal{P} \) is a necessary condition**

We show that Property \( \mathcal{P} \) only depends on transductions.

**Lemma 6.** Let \( T, T' \in \text{ZNFT} \). If \( T \models \mathcal{P} \) and \( T \equiv T' \) then \( T' \models \mathcal{P} \).

**Proof:** Consider two loops \((i_1, j_1), (i_2, j_2)\) as in Property \( \mathcal{P} \) in a run of \( T' \) on some word \( u \). They induce a decomposition of \( u \) as \( u = u_1u_2u_3u_4u_5 \) where \( u_2 = u[i_1 \ldots (j_1 - 1)] \) and \( u_4 = u[i_2 \ldots (j_2 - 1)] \), with \( u_1u_2^k u_3 u_4^l u_5 \in \text{dom}(T') \) for all \( k, l \geq 0 \).

As \( T \) is equivalent to \( T' \) and has finitely many states, there exist iterations of the loops on \( u_2 \) and \( u_4 \) which constitute loops in \( T \) on powers of \( u_2 \) and \( u_4 \). Formally, there exist integers \( d_1, e_1, h_1, d_2, e_2, h_2 \) with \( e_1, e_2 > 0 \) such that \( T \) has a run \( \rho \) on the input word \( u_1u_2^d_1 u_2^e_1 u_3 u_4^d_2 u_4^e_2 u_5 \) which contains a loop on the input subwords \( u_2^d \) and \( u_4^e \).

We conclude easily by using the fact that \( T \models \mathcal{P} \).

**As a consequence, we obtain that Property \( \mathcal{P} \) is a necessary condition for NFT-definability.**

**Lemma 7.** Let \( T \in \text{ZNFT} \). If \( T \) is NFT-definable, then \( T \models \mathcal{P} \).

**Proof:** Let \( T' \) be an NFT equivalent to \( T \). It is easy to turn \( T' \) into a ZNFT \( T'' \) that performs two additional backward and forward passes which output \( \epsilon \). Consider two loops \((i_1, j_1)\) and \((j_1, j_2)\) in a run of \( T'' \), and let us write the output of this run as depicted on Fig. 2. These loops are also loops of \( T' \), and thus we can define \( \beta_1 \) (resp. \( \beta_2 \), \( \beta_3 \) and \( \beta_4 \)) as \( x_0 \) (resp. \( x_1 \), \( x_2 \) and \( x_3 \)), and \( f, g \) as the constant mappings equal to \( \epsilon \). Hence \( T'' \models \mathcal{P} \), and we conclude by Lemma 6.

**B. From ZNFT to \( \epsilon \text{ZNFT} \)**

The goal is to devise a procedure that tests whether the first and second passes (forward and backward) of the run can be done with a single forward pass, and constructs an NFT that realizes this single forward pass. Then, in order to obtain an \( \epsilon \text{ZNFT} \), it suffices to replace the first pass of \( T \) by the latter NFT and add a backward pass that just comes back to the beginning of the word and outputs \( \epsilon \) all the time. The procedure constructs an \( \epsilon \text{ZNFT} \), and tests whether it is equivalent to \( T \). It is based on the following key property that characterizes the form of the output words of the two first passes of any ZNFT satisfying \( \mathcal{P} \). Intuitively, when these words are long enough, they can be decomposed as words whose primitive roots are conjugate.

**Definition 3 (\( \mathcal{P}_1 \)-property).** Let \( T \in \text{ZNFT} \) with \( m \) states, and let \((u, v) \in R(T)\) where \( u \) has length \( n \). Let \( K = 2 \cdot o \cdot m^3 \cdot |\Sigma| \) where \( o = \max \{|v| \mid (p, a, v, q, m) \in \Delta\} \). The pair \((u, v)\) satisfies the property \( \mathcal{P}_1 \), denoted by \((u, v) \models \mathcal{P}_1 \), if for all accepting runs \( \rho \) on \( u \), there exist a position \( 1 \leq \ell \leq n \) and \( w, w', t_1, t_2, t_3 \in \Sigma^* \) such that \( v = w_{t_1, t_2, t_3}w' \) and:

\[
\begin{align*}
\text{out}_1[1, \ell] &= w & \text{out}_2[1, \ell] &= t_3 & \text{out}_1[\ell, n] \text{out}_2[\ell, n] &= t_{12}^* \text{out}_3[1, n + 1] &= w' \\
|t_i| &\leq 2K, \forall i \in \{1, 2, 3\}
\end{align*}
\]

This decomposition is depicted in Fig. 4. \( T \) satisfies property \( \mathcal{P}_1 \), denoted \( T \models \mathcal{P}_1 \), if all \((u, v) \in R(T)\) satisfy it.
 Proposition 4. Let $T \in \text{ZNFT}$. If $[T] \models \mathcal{P}$, then $[\mathcal{T}] \models \mathcal{P}$. 

Proof: • If $\text{out}_2[1, n - 1] \leq K$, then clearly, it suffices to take $\ell = n, t_1 = \text{out}_2[n - 1, n], t_2 = \varepsilon, t_3 = \text{out}_2[0, n - 1], w = \text{out}_1[1, n]$ and $w' = \text{out}_3[1, n + 1]$. 

• Otherwise, $\text{out}_2[1, n - 1] > K$. Therefore $u$ is of length $2, m^3 |\Sigma|$ at least and there exists a (non-empty) loop $(i, j)$ in $\rho$. We can always choose this loop such that $|\text{out}_2[i, j]| \leq K$ and $1 \leq |\text{out}_2[i, j]| \leq K$.

The loop partitions the input and output words into factors that are depicted in Fig. 5 (only the first two passes are depicted). Formally, let $u_0 = u_1 u_2 u_3$ such that $u_2 = u[i \ldots (j - 1)]$. Let $x_0 = \text{out}_1[i, j], v_1 = \text{out}_1[i, j], x_1 = \text{out}_1[j, n]\text{out}_2[j, n], v_2 = \text{out}_2[i, j], x_2 = \text{out}_1[1, i], x_3 = \text{out}_3[1, i], v_3 = \text{out}_3[i, j]$ and $x_4 = \text{out}_3[j, n + 1]$.

In particular, we have $|x_2| \leq K, 1 \leq |v_2| \leq K$ and $x_0 v_1 x_1 v_2 x_2 x_3 v_4 x_4 \in T(u)$. Since $(i, j)$ is a loop we also get $x_0 v_1^k x_1 v_2^k x_2 x_3 v_4^k x_4 \in T(u_1 u_2 u_3)$ for all $k \geq 0$. We then distinguish two cases:

1) If $v_1 \neq \emptyset$. We can apply Property $\mathcal{P}$ by taking the second loop empty. We get that for all $k \geq 0$

$$f(k)x_0 v_1^k c' x_1 v_2^k c' x_2 x_3 v_3^k c' x_4 g(k) = \beta_1 v_1^k \beta_2 \beta_3$$

where $f, g : \mathbb{N} \rightarrow \Sigma^*, c \in \mathbb{N}_{>0}, c' \in \mathbb{N}$, and $\beta_1, \beta_2, \beta_3 \in \Sigma^*$. Since the above equality holds for all $k \geq 0$, we can apply Lemma 1 and we get $\mu(v_1) \sim \mu(\beta_2)$ and $\mu(\beta_2) \sim \mu(v_2)$, and therefore $\mu(v_1) \sim \mu(v_2)$. So there exist $x, y \in \Sigma^*$ such that $v_1 \in (x y)^*$ and $v_2 \in (y x)^*$. One can show (see [24]) that $v_1 x_1 v_2 \in x (y x)^*$. Then it suffices to take $\ell = i, w = x_0, t_1 = x, t_2 = y x_3$ and $t_3 = x_2 x_3$.

2) The second case ($v_1 = \emptyset$) is more complicated as it requires to use the full Property $\mathcal{P}$, using two non-empty loops. First, we distinguish two cases whether $|\text{out}_1[j, n]| \leq K$ or not. For the latter case, we identify a second loop and then apply Property $\mathcal{P}$. 

Construction of an $\varepsilon$ZNFT from a ZNFT We construct an $\varepsilon$ZNFT $T'$ from a ZNFT $T$ such that $R(T') = \{(u, v) \in R(T) | (u, v) \models \mathcal{P}_1\}$. Intuitively, the main idea is to perform the two first passes in a single forward pass, followed by a non-producing backward pass, and the final third pass is exactly as $T$ does. Therefore, $T'$ guesses the words $t_1, t_2$ and $t_3$ and makes sure that the output $v$ is indeed of the form characterized by $\mathcal{P}_1$. This can be done in a one-way fashion while simulating the forward and backward passes in parallel and by guessing non-deterministically the position $\ell$. In addition, the output mechanism of $T'$ exploits the special form of $v$: the idea is to output powers of $t_2$ while simulating the two first passes.

First, let us describe how $T'$ simulates the forward and backward passes in parallel during the first forward pass. It guesses both the state of the backward pass, and the current symbol (this is needed as the symbol read by the backward transition is the next symbol). The first state ($q'$) guessed for the backward pass needs to be stored, as the last (forward) pass should start from $q'$. The transducer can go from state $(p, q, \sigma)$ to state $(y', q', \sigma')$ if the current symbol is $\sigma$ and there is a (forward) transition $(p, \sigma, x, p', +1)$ and a (backward) transition $(q', \sigma', y, q, -1)$. Therefore if $Q$ is the set of states of $T$, $T'$ uses, on the first pass, elements of $Q \times Q \times \Sigma$ in its states. The transducer $T'$ can non-deterministically decide to perform the backward and non-producing backward pass whenever it is in some state $(q, q, \sigma)$ and the current symbol is $\sigma$. This indeed happens precisely when the forward and backward passes are in the same state $q$. If the current symbol is not the last of the input word, then the whole run of $T'$ is not a $\varepsilon$-motion and therefore it is not accepting.

Second, we describe how the $\varepsilon$ZNFT $T'$, with the guess of $t_1, t_2, t_3$, verifies during its first forward pass that the output has the expected form, and how it produces this output. During the first pass, $T'$ can be in two modes: In mode 1 (before the guess $\ell$), $T'$ verifies that the output on the simulated backward pass is $t_3$ and proceeds as $T$ in the first forward pass (it outputs what $T$ outputs on the forward pass). Mode 2 starts when the guess $\ell$ has been made. In this mode, $T'$ first outputs $t_1$ and then verifies that the output of the forward/backward run from and to position $\ell$ is of the form $t_1 t_2$. It can be done by using pointers on $t_1$ and $t_2$. There are two cases (guessed by $T'$): either $t_1$ ends during the forward pass or during the backward pass (using notations of Fig.4, either $t_1$ is a prefix of $x$, or $x$ is a prefix of $t_1$).

In the first case, $T'$ needs a pointer on $t_1$ to make sure that the output of $T$ in the forward pass starts with $t_1$. It also needs a pointer on $t_2$, initially at the end of $t_2$, to make sure that the output of $T$ on the simulated backward pass is a suffix of $t_2^*$ (the pointer moves backward, coming back to the last position of $t_2$ whenever it reaches the first position of $t_2$). Once the verification on $t_1$ is done, $T'$ starts, by using a pointer initially at the first position in $t_2$, to verify that the output of $T$ in the forward pass is a prefix of $t_2$. Once the forward and the simulated backward passes merge, the two pointers on $t_2$ must be at the same position, otherwise the run is rejected.

During this verification, $T'$ also has to output a power of $t_2$ (remind that it has already output $t_1$). However the transitions of $T'$ may not output exactly one $t_2$, nor a power of $t_2$, but may cut $t_2$ before its end. Therefore $T'$ needs another pointer $h$ to know where it is in $t_2$. Initially this pointer is at the first position of $t_2$ ($h = 1$). Suppose that $T'$ simulates $T$ using the (forward) transition $(p, \sigma, x, p', +1)$ and the (backward) transition $(q', \sigma', y, q, -1)$. If this step occurs before the end of $t_1$, then $T'$ outputs $t_2^h[h \ldots (h + |y|)] (t_2^h$ is the infinite concatenation of $t_2^h)$, and the pointer $h$ is updated to $1 + ((h + |y| - 1) \mod |t_2^h|)$. Otherwise, $T'$ outputs $t_2^h[h \ldots (h + |x| + |y|)]$
and \( h \) is updated to \( 1 + ((h + |x| + |y| - 1) \mod |t_2|) \).

The second case (when \( T' \) guesses that \( t_1 \) ends during the backward pass) is similar. \( T' \) has to guess exactly the position in the output where \( t_1 \) ends. On the first pass it verifies that the output is a prefix of \( t_1 \), and on the simulated backward pass, it checks that the output is a suffix of \( t_2 \) (and outputs as many \( t_2 \) as necessary, like before), until the end of \( t_1 \) is guessed to occur. From that moment it enters a verification mode on both passes.

The main property of this construction is that no wrong output words are produced by \( T' \), due to the verification and the way the output words are produced, i.e. for all \( (u, v) \in R(T') \), we have \( (u, v) \in R(T) \).

**Proposition 5.** Let \( T \in \text{ZNFT} \). \( R(T') = \{(u, v) \in R(T) | (u, v) \models P_1\} \).

**Lemma 8.** Let \( T \in \text{ZNFT} \). If \( T \models P \), then \( T \) is equivalent to the \( \epsilon \text{ZNFT} T' \). Moreover, the latter is decidable.

**Proof:** If \( T \models P \), then by Proposition 4, \( T \models P_1 \). Therefore by Proposition 5, \( T \) and \( T' \) are equivalent.

We know that \( R(T') \subseteq R(T) \), and since \( T \) and \( T' \) are both functional, they are equivalent iff \( \text{dom}(T) \subseteq \text{dom}(T') \). Both domains can be defined by NFAs. Those NFAs simulate the three passes in parallel and make sure that those passes define a \( \epsilon \)-transition. Therefore testing the equivalence of \( T \) and \( T' \) amounts to test the equivalence of two NFAs.

**C. From \( \epsilon \text{ZNFT} \) to \( \text{NFT} \)**

We have seen how to go from a \( \text{ZNFT} \) to an \( \epsilon \text{ZNFT} \). We now briefly sketch how to go from an \( \epsilon \text{ZNFT} \) to a (functional) \( \text{NFT} \). Given an \( \epsilon \text{ZNFT} T' \), we define an \( \text{fNFT} T'' \) such that \( T' \) and \( T'' \) are equivalent as soon as \( T' \models P \). The ideas are very similar to the previous construction therefore we do not give all the details here.

We exhibit a property on the form of output words produced by an \( \epsilon \text{ZNFT} \) that verifies \( P \). Intuitively, apart from the beginning of the first pass, and the end of the second pass, if the two passes produce long enough outputs, then these outputs can be decomposed so as to exhibit conjugate primitive roots.

**Definition 4 (\( P_2 \)-property).** Let \( T' \in \text{ZNFT} \) with \( m \) states, and let \( (u, v) \in R(T') \) where \( u \) has length \( n \). Let \( K = 2m^{3n}|\Sigma| \) where \( o = \max\{|v| | (p, a, v, q, m) \in \Delta\} \). The pair \( (u, v) \) satisfies the property \( P_2 \), denoted by \( (u, v) \models P_2 \), if for all accepting runs \( \rho \) on \( u \), there exist two positions \( 1 \leq \ell_1 \leq \ell_2 \leq n \) and \( w, w', t_1, t_2, t_3 \in \Sigma^* \) such that:

\[
\begin{align*}
&\text{out}_1[1, \ell_1] = w & |t_i| \leq 3K, \forall i \in \{1, 2, 3\} \\
&\text{out}_3[\ell_2, n+1] = w' & |\text{out}_3[\ell_2, n]| \leq 3K \\
&\text{out}_1[\ell_1, n] \text{out}_3[1, \ell_2] \in t_1 t_2^* t_3 & |\text{out}_3[1, \ell_1]| \leq 3K \\
\end{align*}
\]

This decomposition is depicted in Fig. 6. \( T' \) satisfies property \( P_2 \), denoted \( T' \models P_2 \), if all \( (u, v) \in R(T') \) satisfy it.

The proof of the following proposition uses the same structure and techniques as that of Proposition 4. Using a (long) case analysis, we identify loops in runs, and apply Property \( P \) to show that output words have the expected form.

**Proposition 6.** Let \( T' \in \text{fZNFT} \). If \( T' \models P \), then \( T' \models P_2 \).

We can now sketch the construction of an \( f\text{NFT} T'' \) which recognizes the subrelation of \( T' \) defined as \( \{(u, v) \in R(T') | (u, v) \models P_2\} \). Again, the construction is rather similar and uses the same techniques to that of \( T' \) starting from \( T \).

The transducer \( T'' \) simulates, in a single forward pass, the three passes of \( T' \). Hence it also checks that the run of the \( \text{ZNFT} T' \) it simulates is a \( 3 \)-motion run, which is a semantic restriction of accepting runs of ZNFTs. The \( f\text{NFT} T'' \) also guesses positions \( \ell_1 \) and \( \ell_2 \), and uses three modes accordingly. It also guesses the words \( t_1, t_2 \) and \( t_3 \), and words for \( \text{out}_3[1, \ell_1] \) and \( \text{out}_1[\ell_2, n] \), which are all of bounded length (see Property \( P_2 \)). The output of \( T'' \) is produced according to the mode, using pointers to check the guesses, similarly to \( T' \).

If all the guesses happen to be verified, it outputs the correct output word, otherwise the input word is rejected. As a consequence, \( T'' \) recognizes a subrelation of \( T' \) and thus checking the equivalence of \( T' \) and \( T'' \) amounts to checking the equivalence of their domains (as the two transducers are functional), which is decidable. From Proposition 6 we get:

**Lemma 9.** Let \( T' \in \text{fZNFT} \). If \( T' \models P \), then \( T' \) is equivalent to the \( f\text{NFT} T'' \). Moreover, the latter property is decidable.

**Proof of Lemma 3.** Lemma 7 states that if \( T \) is \( \text{NFT} \)-definable, then \( T \models P \). Conversely, if \( T \models P \), then by Lemma 8, the first construction outputs an equivalent \( \epsilon \text{ZNFT} T' \). By Lemma 6, we have \( T' \models P \). By Lemma 9, the second construction outputs an equivalent \( \text{NFT} T'' \). Therefore \( T \) is \( \text{NFT} \)-definable by \( T'' \). In order to decide whether \( T \models P \), it suffices to construct \( T' \), check that \( T \) and \( T' \) are equivalent, and then construct \( T'' \) and check whether \( T' \) and \( T'' \) are equivalent. Both problems are decidable by Lemma 8 and 9.

**V. DISCUSSION**

**Complexity.** The procedure to decide \((f\text{NFT}, \text{NFT})\)-definability is non-elementary exponential time and space. This is due to the \( \text{ZNFT} \)-to-\( \text{NFT} \) construction which outputs an \( \text{NFT} \) of doubly exponential size. Indeed, the first step of this construction transforms any \( \text{ZNFT} \) with \( n \) states into an \( \epsilon \text{ZNFT} \) with at least \( |\Sigma|^{4n^3}|\Sigma| \) states, as the \( \epsilon \text{ZNFT} \) has to guess words of length \( 4n^3|\Sigma| \), where \( o \) is the maximal length of an output word of a transition. The \( \text{ZNFT} \)-to-\( \text{NFT} \) construction also outputs an exponentially bigger transducer.
Therefore the squeeze operation outputs a transducer which is doubly exponentially larger. Since this operation has to be iterated \( N^2 \) times in the worst case, where \( N \) is the number of states of the initial \( 2\text{NFT} \), this leads to a non-elementary procedure. On the other hand, the best lower bound we have for this problem is PSpace (by a simple proof that reduces the emptiness problem of the intersection of \( n \) DFAs is given in [24]).

**Succinctness** It is already known that 2DFA\( s \) are exponentially more succinct than NFAs [27]. Therefore this result carries over to transducers, already for transducers defining identity relations on some particular domains. However we show here a stronger result: the succinctness of \( 2\text{NFT}s \) also comes from the transduction part and not only from the domain part. We can indeed exhibit a family of NFT-definable transductions \( (R_n)_{n} \), that can be defined by 2DFT\( s \) that are exponentially more succinct than their smallest equivalent NFT, and such that the family of languages \( (\text{dom}(R_n))_n \) does not show an exponential blow up between 2DFA\( s \) and NFAs. For all \( n \geq 0 \), we define \( R_n \) whose domain is the set of words \( \#_u \# \) for all \( u \in \{a,b\}^* \) of length \( n \), and the transduction is the mirror transduction, i.e. \( R_n(\#_u \#) = \#_u \# \). Clearly, \( R_n \) is definable by a 2DFT with \( O(n) \) states that counts up to \( n \) the length of the input word by a forward pass, and then mirrors it by a backward pass. It is also definable by an NFT with \( O(2^n) \) states: the NFT guesses a word \( u \) of length \( n \) (so it requires \( O(2^n) \) states), outputs its reverse, and then verifies that the guess was correct. It is easy to prove that any NFT defining \( R_n \) needs at least \( 2^n \) states by a pumping argument. On the other hand, the domain of \( R_n \) can be defined by a DFA with \( O(n) \) states that counts the length of the input word up to \( n \). Note that the alphabet does not depend on \( n \).

**Further Questions** We have shown that \( (2\text{NFT},\text{NFT}) \)-definability is decidable, however with a non-elementary procedure. We would like to characterize precisely the complexity of this problem. Our procedure works for functional 2NFT\( s \), which are equivalent to 2DFT\( s \). Therefore we could have done our proof directly for 2DFT\( s \). However (functional) non-determinism was added with no cost in the proof so we rather did it in this more general setting. The extension of our results to relations instead of functions is still open.

Our proof is an adaptation of the proof of Rabin and Scott [3] to transducers. Alternative constructions based on the proofs of Shepherdson [4] or Vardi [17], and alternative models such as streaming string transducers [10] or MSO transformations [8], [9], could lead to better complexity results or refined results. In particular, we believe that our results are highly related to the problem of minimizing the number of variables in a streaming string transducer.

Finally, we plan to study extensions of our results to infinite string transformations, defined for instance by streaming string transducers [28], and to tree transformations, following our initial motivation from XML applications.

**Acknowledgements** We warmly thank Sebastian Maneth and Julien Tierny for interesting discussions.

**References**


